CIT 591 Introduction to Software Development

Module 6: File Input and Output (File I/O)

Module Learning Objectives

- Move from interaction via the console to using data stored in files
- Analyze text data from a file
- Control access to the instance variables and methods of a class

Module Glossary

- **Access Modifiers:** Keywords added before an instance variable declaration or a method declaration that determine whether or not the instance variable or method can be accessed outside of the class where it is being defined.
- **Buffer:** Since writing data to a file, which is typically stored on disk, is usually slow, Java buffers a larger chunk of the data into a separate area of memory - the buffer. Then, as system resources free up, data will be transferred from the buffer to the disk.
- **Encapsulation:** One of the main ideas in object oriented programming, encapsulation is the idea of grouping the data and methods that work on that data into one unit - a class.
- **`flush`:** A lot of the inbuilt file writer classes in Java buffer the content to be written before it actually gets written to the destination. The flush command ensures that the buffer gets flushed and everything that was asked to be written does get written at the point of the flush. Full details >
- **Getters and Setters:** Methods used to access private instance variables. Also known as **accessors and mutators**.
- **`indexOf`:** An inbuilt method used for obtaining the index in a string where a character can be found. For example `“computer”.indexOf(“o”)` will give you 1.
- **`Integer.parseInt`:** An inbuilt method used for converting a string to an integer.
- **`public`**: An access modifier that allows classes outside of this current one to access modify the variable or call the method.
- **`private`**: An access modifier that only allows the current class to work with the given instance variable or method.

Module Resources

- **Textbook Readings:**
  - 2.4 Constructing Objects
  - 2.5: Accessor and Mutator Methods
  - 11.1: Reading and Writing Text Files
  - 11.2: Text Input and Output
- **Websites:**
  - indexOf (Oracle Docs)
  - parseInt (Oracle Docs)
  - Default Values in Java (Oracle Docs)
  - Access Control in Java

Key Concepts & Examples

Constructors: Prior to this week, we've been using Java's default constructor using the new keyword. Now you can write your own constructor, which we recommend doing whenever you create a new class. This makes the creation of objects convenient and it encapsulates creation of the object into one method.

- Examples used in the video: Continuing the car example; revisiting the chessPiece example

Access Modifiers: Access modifiers describe the keywords public and private that we have been seeing - these determine whether a variable or method can be accessed outside the class where it has been defined. Once an object is made, its instance variable values should only be changed via methods. The rule of thumb is to make all instance variables private and all methods public.

If you need for a private instance variable to be set or read from an external class, you can create getters and setters.

- Examples used in the video: our BankAccount class has a private variable **balance** and a public method **deposit**; updating our Car class to have private variables and public methods

Reading Data from a File: We can use the same Scanner from week 1 to read text from an external file (e.g. a CSV dataset, text file, etc.) and store it in memory. Remember to import `java.io.*` at the top to be able to use the File class.

Don't worry if Eclipse reports an unhandled exception when you import the file - use the suggested try/catch block for now and we will learn more about this in our module on Exception Handling.

- Example used in the video: importing a simple text file and counting the number of words

Writing Data to a File: Reading and writing to files combine to form one of the most useful tools for Java programming and form the foundation of data science. In this lecture, we read in a data file and saved the data into an ArrayList, looped through the data to run some calculations, then used the PrintWriter to print the calculation straight into the file.

Things to watch out for: Printwriter temporarily stores whatever it's about to write into a staging area called a buffer. So in order to see the result immediately, we have to call a method called **flush** to flush from the buffer to the actual file.

- Examples used in the video: scores for a test - looping over the values to calculate the mean and standard deviation